CS265, Winter 2022

Class 3: Agenda, Questions, and Links

1 Warm-Up

Let $S_r = \{x \in \mathbb{Z}_n^* : x^r = \pm 1 \mod n\}$. Is S_r a group?

2 Announcements

- HW1 is due Wednesday!
- Starting today we will release quiz grades/answers at the *start* of class, so that you can see the answers and discuss them when you go to your groups to talk about the material.

3 Questions?

Any questions from the minilectures or the quiz? (Group Theory 101; Primality Testing)

- Go into small groups and ask each other your questions.
- Ask any questions that the group can't resolve in the question box. Upvote each others' questions!

4 Miller-Rabin Algorithm

[A bit of lecture to introduce the Miller-Rabin Algorithm. Summary below.]

The main idea behind the Miller-Rabin algorithm is the fact (which is not obvious) that:

- If n is prime, then there are exactly two square roots of 1 in $\mathbb{Z}_n^*$, $+1$ and -1.
- If n is an odd composite number, not the power of a prime, then there are *more* than two square roots of 1 in $\mathbb{Z}_n^*$.

The following Proposition is true, check out the lecture notes (or think about it yourself if you finish the group work early!) for more details.

Proposition 1. *If n is prime, then there are exactly two square roots of* 1 *in $\mathbb{Z}_n^*$,* $+1$ *and* -1.

Consider the following way to generate a list of numbers, using n. (It should not be obvious at this point why we are doing this).

Procedure for generating some numbers

- Write $n-1 = 2^k m$ where m is odd.
- Choose $x \in \{1, \ldots, n-1\}$ uniformly at random.
- Consider the list of numbers $x^m, x^{2m}, x^{2^2 m}, \ldots, x^{2^k m} \mod n$.

In the following group work, we'll play around with some examples.

4.1 Group work: Exploring this list of numbers

Group Work

First, introduce yourselves to each other. Then work through the following questions. It might be a good idea to read each question silently and think about it on your own before coming together to confer.

In this breakout room session, we'll investigate the list of numbers generated according to the algorithm above.

As you go through the questions, please click 1-5 on the green poll to indicate when you finish a part! (e.g., clicking "1" means you are done with part 1). Everyone in the room can do this).

1. Figure out how to generate these lists automatically. Here are ways you can do that.
   - Go to

     https://web.stanford.edu/~marykw/CS265Class3.html

     This just runs a python script that takes x and n and generates this list.
   - If you want to do it yourself, there's some python code at the end of this document that you can copy and paste.

   Make sure that you can replicate the example we saw with $x = 3$, $n = 21$. You should get $12, 18, 9$.
2. Suppose that n is prime. What should you get as the *last* number generated in this list? (Hint, Fermat's little theorem). Verify your answer by trying it out.
3. Suppose that n is prime. What are the possible answers you could get as the *second-to-last* number generated in this list? (Hint, Proposition 1). Verify your answer by trying it out.

   (Note: If you see "$n-1$" show up in the widget, remember that this is the same as "-1" mod n.)
4. Suppose that $n = 561$. (Recall that this is the first Carmichael number—so it's not prime, but for any x with $gcd(x, n) = 1$, we have $x^{n-1} = 1 \mod n$.)

- Choose $x = 23$. What do you get? Why does this answer prove to you that n is not prime? (Hint, previous question).
- Choose $x = 13$. What do you get? Why does this answer prove to you that n is not prime? (Hint, Prop. 1).
- Choose $x = 63$. What do you get? Why does this answer prove to you that n is not prime? (Hint, Fermat's little theorem).
- Choose $x = 458$. Does this answer prove to you that n is not prime? Why or why not?

5. Come up with a candidate randomized algorithm to test primality based on these observations. You can try out some other examples to test it.

   Don't forget to update the poll!

6. If you have time, try to prove that your candidate randomized algorithm works!

4.2 Developing the Miller-Rabin Test

[A bit of lecture]

4.3 Group Work: Analyzing the Miller-Rabin Test

Now that we know the Miller-Rabin Test, we want to analyze it. The relevant part of the test[1] is:

Relevant part of Miller-Rabin Test

- Generate the list of k numbers as above.
- If the last number is not 1, output "composite!"
- If there's a number that's *not* equal to ± 1, but the next number is equal to 1, output "composite!"
- Otherwise, output "prime!"

[On slides, we will define the following set and make the following claims.]
Recall that we defined

$$S = \left\{ y \in \mathbb{Z}_n^* \, : \, y^{2^b m} = \pm 1 \mod n \right\},$$

where b is the largest value $i < k$ so that there exists an $x^{2^i m} = \pm 1 \mod n$. We made the following claims:

- Claim 1: For any x so that the algorithm says "Prime!", $x \in S$.
- Claim 2: S is a subgroup of $\mathbb{Z}_n^*$.

[1] See lecture notes for the actual algorithm

- Claim 3: If n is odd, composite, and not a prime power, $S \neq \mathbb{Z}_n^*$.

Group Work

Assuming Claims 1,2, and 3, prove the following:

If n is odd, composite, and not a prime power, the algorithm above says "Composite!" with probability at least $1/2$.

(We have already waved our hands about this in class. What you're supposed to be doing now is explaining to each other. Try to write down a formal proof to see how all the pieces fit together.)

Hint: Recall that if S is a proper subgroup of $\mathbb{Z}_n^*$, then by Lagrange's theorem, $|S| \leq \frac{|\mathbb{Z}_n^*|}{2}$.

Once you are done **please click "Yes!" in the Yes/No poll.**

If you have time, think about hte following:

- Prove Proposition 1. (Hint: use the fact that $\mathbb{Z}_n^*$ is cyclic).
- Prove Claim 3. (Disclaimer: this is tricky!) (Hint: by the assumptions on n, we can write $n = s \cdot t$ where s and t are relatively prime. The *Chinese Remainder Theorem* implies the following. For any s and t that are relatively prim that for any $x \in \mathbb{Z}_n^*$, there exists a $y \in \mathbb{Z}_n^*$ so that $y = x \mod s$ and $y = 1 \mod t$. Show that such a y satisfies $y \in \mathbb{Z}_n^*$ but $y \notin S$.)

5 Useful Code

Python code to generate the list of numbers, given x and n:

```
def generateList(x,n):
if n <= 1:
return []
        # generate k and m
        k = 0
        m = n-1
        while m % 2 == 0:
                k += 1
                m = m/2
        # now generate x^m, x^(2m), x^(4m), ..., x^(2^k m) mod n
        ret = []
        for i in range(k+1):
                y = x**int(m) % n
                for j in range(i):
```

```
                y = y**2 % n
        ret.append(int(y))
return ret
```